设计模式之单例

- 定义

  保证一个类仅有一个实例, 并提供一个访问它的全局访问点

- 简单的单例

```
let Singleton = function(name){
    this.name = name
    this.instance = null
}

Singleton.getInstance = function(name){
    if(!this.instance) {
        this.instance = new Singleton(name)
    }
    return this.instance
}

let tom = Singleton.getInstance('tom')
let lusy = Singleton.getInstance('lusy')

tom == lusy // true
```

- 使用代理模式实现单例

```
let proxyCreateSingleton = (function(){
    let instance = null
    return function(name){
        if(!instance) instance = new Singleton(name)
        return instance
    }
})()

let tom = proxyCreateSingleton('tom')
let lusy = proxyCreateSingleton('lusy')

tom == lusy // true
```

- 惰性单例(需要时才创建)

```
function createDiv(){
    return document.createElement('div')
}

let getSingleton = function(fn){
    let instance = null
    return function(){
        if(!instance) instance = fn.apply(this, arguments)
        return instance
    }
}
```

```
let createSingleDiv =  getSingleton(createDiv)

createSingleDiv() == createSingleDiv() // true
```

- 单例模式的使用
    - 当创建对象十分频繁时, 对象开销较大时可以考虑使用
    - 在登录, 注册弹框是只希望永远只有一个弹窗是可以考虑使用